Applications of Discrete Mathematics to the Analysis of Algorithms

Conrado Martínez

Univ. Politècnica de Catalunya, Spain

May 2007

Goal

- Given some algorithm A taking inputs from some set $\mathcal{I}$, we would like to analyze the performance of the algorithm as a function of the input size (and possibly other parameters).

Why?

- To predict the resources (time, space. . . .) that the algorithm will consume
- To compare algorithm A with competing alternatives
- To improve the algorithm, by spotting the performance bottlenecks
- To explain observed behavior

Why?

- To predict the resources (time, space. . . .) that the algorithm will consume
- To compare algorithm A with competing alternatives
- To improve the algorithm, by spotting the performance bottlenecks
- To explain observed behavior

Why?

- To predict the resources (time, space. . . .) that the algorithm will consume
- To compare algorithm A with competing alternatives
- To improve the algorithm, by spotting the performance bottlenecks
- To explain observed behavior

Why?

- To predict the resources (time, space. . . .) that the algorithm will consume
- To compare algorithm A with competing alternatives
- To improve the algorithm, by spotting the performance bottlenecks
- To explain observed behavior

- The performance $\mu : \mathcal{I} \to \mathbb{N}$ depends on each particular instance of the input
- We have to introduce some notion of size: $|\cdot| : \mathcal{I} \to \mathbb{N}$; we may safely assume that each $\mathcal{I}_n = \{x \in \mathcal{I} \,|\, |x| = n\}$ is finite
- Worst-case:

$$\mu^{[\text{worst}]}(n) = \max\{\mu(x) \,|\, x \in \mathcal{I}_n\}$$

- To analyze "typical behavior" or the performance of randomized algorithms, we have to assume some probabilistic distribution on the input and/or the algorithm's choices; hence, we consider the performance as a family of random variables $\{\mu_n\}_{n\geq 0}$; $\mu_n : \mathcal{I}_n \to \mathbb{N}$
- Average-case:

$$\mu^{[\text{avg}]}(n) = \mathbb{E}[\mu_n] = \sum_{k\geq 0} k\, \mathbb{P}[\mu_n = k]$$

When we assume uniformly distributed inputs

$$\mathbb{P}[x] = \frac{1}{\#\mathcal{I}_n}, \text{for all } x \in \mathcal{I}_n$$

our problem is one of counting, e.g.,

$$\mathbb{E}[\mu_n] = \frac{\sum_{x \in \mathcal{I}_n} \mu(x)}{\#\mathcal{I}_n}$$

One of the most important tools in the analysis of algorithms are generating functions:

$$A(z,u) = \sum_{n\geq 0}\sum_{k\geq 0} \mathbb{P}[\mu_n = k]\, z^n u^k$$

For the uniform distribution

$$A(z,u) = \frac{\sum_{n\geq 0}\sum_{k\geq 0} a_{n,k} z^n u^k}{\sum_{n\geq 0} a_n z^n} = \frac{B(z,u)}{B(z,1)}$$

with $a_{n,k} = \#\{x \in \mathcal{I} \,|\, |x| = n \wedge \mu(x) = k\}$ and $a_n = \#\mathcal{I}_n$

The equations before can be expressed symbolically

$$B(z,u) = \sum_{x \in \mathcal{I}} z^{|x|} u^{\mu(x)}$$

The ratio of the n–th coefficients of $B(z,u)$ and $B(z,1)$ is the probability generating function of μ_n

$$p_n(u) = \sum_{k \geq 0} \mathbb{P}[\mu_n = k]\, u^k = \frac{[z^n]B(z,u)}{[z^n]B(z,1)}$$

Taking derivatives w.r.t. u and setting $u = 1$ we get the expected value, second factorial moment, ...

$$A^{(r)}(z) = \left.\frac{\partial^r A(z,u)}{\partial u^r}\right|_{u=1}$$
$$= \sum_{n\geq 0} \mathbb{E}[\mu_n{}^{\underline{r}}]\, z^n$$

For example,

$$\mathbb{V}[\mu_n] = \mathbb{E}\left[\mu_n{}^{\underline{2}}\right] - \mathbb{E}[\mu_n]^2 = [z^n]A^{(2)}(z) - ([z^n]A(z))^2$$

The symbolic method translates combinatorial constructions to functional equations over generating functions.

Example: Consider the counting generating function of a combinatorial class $\mathcal{A}$:

$$A(z) = \sum_{n \geq 0} a_n z^n = \sum_{\alpha \in \mathcal{A}} z^{|\alpha|}$$

If $\mathcal{A} = \mathcal{B} \times \mathcal{C}$ then

$$\begin{aligned} A(z) &= \sum_{\alpha \in \mathcal{A}} z^{|\alpha|} = \sum_{(\beta,\gamma) \in \mathcal{B} \times \mathcal{C}} z^{|\beta|+|\gamma|} = \left(\sum_{\beta \in \mathcal{B}} z^{|\beta|} \right) \left(\sum_{\gamma \in \mathcal{C}} z^{|\gamma|} \right) \\ &= B(z) \cdot C(z) \end{aligned}$$

A dictionary of (labelled) combinatorial constructions and G.F.'s

$\{\epsilon\}$	1
$\{Z\}$	z
$\mathcal{A} + \mathcal{B}$	$A + B$
$\mathcal{A} \times \mathcal{B}$	$A \cdot B$
$\mathrm{Seq}(\mathcal{A})$	$\frac{1}{1-A}$
$\mathrm{Set}(\mathcal{A})$	$\exp(A)$
$\mathrm{Cycle}(\mathcal{A})$	$\log \frac{1}{1-A}$

A trivial example: since a Binary tree is either an empty tree (leaf) or a root together with two Binary (sub)trees, we have

$$\mathcal{B} = \{\epsilon\} + \{Z\} \times \mathcal{B} \times \mathcal{B}$$

Hence the counting GF of Binary trees is

$$B(z) = 1 + zB^2(z)$$

Solving the equation before for $B(z)$ and since $B(0) = b_0 = 1$,

$$B(z) = \begin{cases} \frac{1-\sqrt{1-4z}}{2z} & z \neq 0, \\ 1 & z = 0. \end{cases}$$

Extracting the n-th coefficient of $B(z)$ we find

$$[z^n]B(z) = \frac{\binom{2n}{n}}{n+1}$$

A more sophisticated example arises in the analysis of the number of branch mispredictions made by quicksort (Kaligosi, Martínez, Sanders, 2006).

We have a random permutation σ of $[1..n]$ and we scan it from left to right. Assume $\sigma_1 = k$.

- For $2 < i \leq k$, we say there is a left Brach misprediction whenever $\sigma_{i-1} < k$ and $\sigma_i > k$, or $\sigma_{i-1} > k$ and $\sigma_i < k$.
- For $k \leq j < n$, there is a right BM if $\sigma_{j+1} < k$ and $\sigma_j > k$, or $\sigma_{j+1} > k$ and $\sigma_j < k$.
- Additionally, there is a left BM if $\sigma_2 > k$ and a right BM if $\sigma_n < k$.

We have a random permutation σ of $[1..n]$ and we scan it from left to right. Assume $\sigma_1 = k$.

- For $2 < i \leq k$, we say there is a left Brach misprediction whenever $\sigma_{i-1} < k$ and $\sigma_i > k$, or $\sigma_{i-1} > k$ and $\sigma_i < k$.
- For $k \leq j < n$, there is a right BM if $\sigma_{j+1} < k$ and $\sigma_j > k$, or $\sigma_{j+1} > k$ and $\sigma_j < k$.
- Additionally, there is a left BM if $\sigma_2 > k$ and a right BM if $\sigma_n < k$.

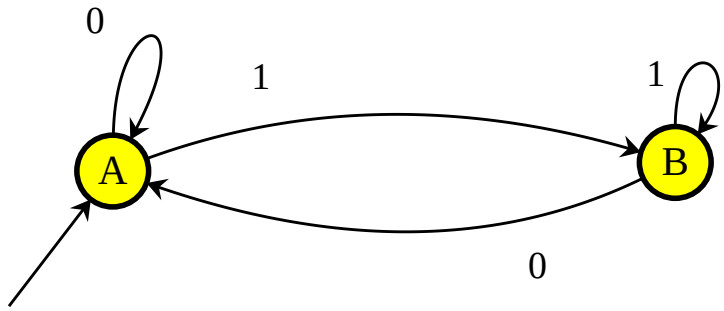
We have a random permutation σ of $[1..n]$ and we scan it from left to right. Assume $\sigma_1 = k$.

- For $2 < i \leq k$, we say there is a left Brach misprediction whenever $\sigma_{i-1} < k$ and $\sigma_i > k$, or $\sigma_{i-1} > k$ and $\sigma_i < k$.
- For $k \leq j < n$, there is a right BM if $\sigma_{j+1} < k$ and $\sigma_j > k$, or $\sigma_{j+1} > k$ and $\sigma_j < k$.
- Additionally, there is a left BM if $\sigma_2 > k$ and a right BM if $\sigma_n < k$.

- We want to know the expected number of branch mispredictions when the choosen pivot is the k-th element of the array.
- We transform the original problem to counting bitstrings. Given a bistring x of length k starting with a 0 and containing $t+1$ 0's, the number of left BMs is the number of times we find a 0 followed by a 1, or a 1 followed by a 0 in x.
- We go from a permutation σ to a bitstring x by setting $x_i = 0$ if $\sigma_i \leq \sigma_1 = k$, and $x_i = 1$ otherwise.

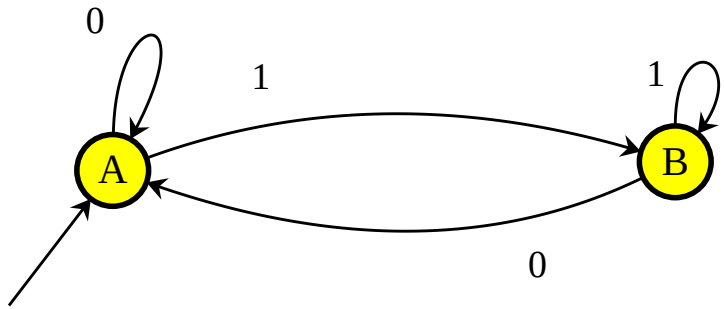
- We want to know the expected number of branch mispredictions when the choosen pivot is the k-th element of the array.
- We transform the original problem to counting bitstrings. Given a bistring x of length k starting with a 0 and containing $t+1$ 0's, the number of left BMs is the number of times we find a 0 followed by a 1, or a 1 followed by a 0 in x.
- We go from a permutation σ to a bitstring x by setting $x_i = 0$ if $\sigma_i \leq \sigma_1 = k$, and $x_i = 1$ otherwise.

- We want to know the expected number of branch mispredictions when the choosen pivot is the k-th element of the array.
- We transform the original problem to counting bitstrings. Given a bistring x of length k starting with a 0 and containing $t+1$ 0's, the number of left BMs is the number of times we find a 0 followed by a 1, or a 1 followed by a 0 in x.
- We go from a permutation σ to a bitstring x by setting $x_i = 0$ if $\sigma_i \leq \sigma_1 = k$, and $x_i = 1$ otherwise.



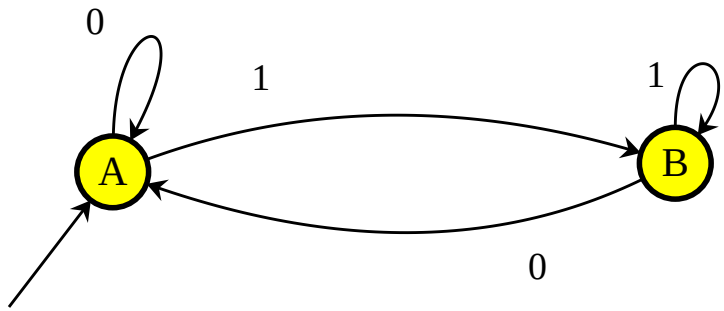
$a_{n,t,r}$ = number of strings of length n with t 0's ending at state A and incurring r BMs

$b_{n,t,r}$ = number of strings of length n with t 0's ending at state B and incurring r BMs

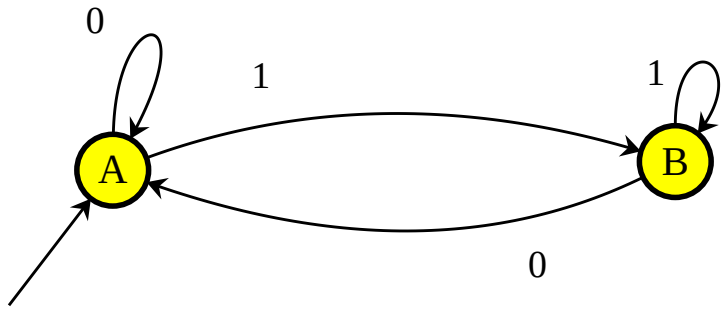


$$A(x,u,z) = \sum_{n,r,t} a_{n,t,r} x^t u^r z^n,$$

$$B(x,u,z) = \sum_{n,r,t} b_{n,t,r} x^t u^r z^n,$$



$$A = 1 + xzA + xuzB,$$
$$B = zB + zuA.$$



$$C = A + B = \frac{1 - z + uz}{(1 - z)\left(1 - uy - \frac{xu^2z^2}{1-z}\right)}$$

Take derivatives with respect to u, set $u = 1$, and extract the coefficient $R_{k,t}$ of $z^{k-1}y^t$ in C; then multiply by the number of random permutations producing a bitstring with $t + 1$ 0's among the first k bits and sum for all t.

Another important set of techniques comes from complex variable analysis.
Under suitable technical conditions, if $F(z)$ is analytic in a disk $D = \{z \in \mathbb{C} \mid |z| < 1\}$ and has a single dominant singularity at $z = 1$ then

$$F(z) \sim G(z) \implies [z^n]F(z) \sim [z^n]G(z)$$

This is one of the useful transfer lemmas of Flajolet and Odlyzko (1990). Many other similar results are extremely useful when computing asymptotic estimates for the n-th coefficient of a generating function.

In recent years, complex analysis techniques and perturbation theory have been used to prove powerful results such as Hwang's Quasi-power theorem, which allows one to prove the convergence in law to a Gaussian distribution of many combinatorial parameters in strings, permutations, trees, etc., as well as local limits and the speed of convergence.

Another example is motivated by the analysis of a combinatorial algorithm that shuffles two trees.

Given two binary trees T_1 and T_2 their shuffle or intersection is defined as follows

$$T_1 \cap T_2 = \begin{cases} \square & \text{if } T_1 = \square \text{ or } T_2 = \square, \\ \circ(L_1 \cap L_2, R_1 \cap R_2) & \text{if } T_i = \circ(L_i, R_i) \end{cases}$$

Let

$$S(x,y) = \sum_{(T_1,T_2)\in\mathcal{B}\times\mathcal{B}} \Pr(T_1,T_2)\,|T_1\cap T_2|x^{|T_1|}y^{|T_2|}$$

The coefficient of $x^m y^n$ in $S(x,y)$ is the average size of the intersection of a pair of trees with sizes m and n, resp.

If we assume independently drawn trees then

$$\Pr(T_1, T_2) = \Pr(T_1) \cdot \Pr(T_2)$$

If we assume that the total size of (T_1, T_2) is n and all possible partitions are equally likely (Binary search tree probability model)

$$\Pr(T_1, T_2) = \frac{\Pr(T_1)\Pr(T_2)}{|T_1| + |T_2| + 1}$$

Furthermore,

- In the uniform probability model $\Pr(T) = 1/b_n$, with $n = |T|$.
- In the binary search tree model

$$\Pr(T) = \begin{cases} 1 & \text{if } T = \square, \\ \frac{\Pr(L)\Pr(R)}{|T|} & \text{if } T = \circ(L, R). \end{cases}$$

In the binary search tree probability model, the symbolic method yields the hyperbolic PDE that $S(x,y)$ satisfies

$$\frac{\partial^2 S(x,y)}{\partial x \partial y} = \frac{1}{(1-x)^2(1-y)^2} + 2\frac{S(x,y)}{(1-x)(1-y)},$$

and $S(x,0) = S(0,y) = 0$.

If we consider

$$\psi(z) = \frac{1}{z} \int_0^z S(t,t)\, dt$$

then $[z^n]\psi(z)$ is the average size of the intersection of a pair of trees of total size n drawn according to the Binary search tree probability model.

A delicate analysis of the solution of the PDE for $S(x, y)$ shows that

$$\psi(z) \sim (3 + 2\sqrt{2}) \cdot J_0\left(2\sqrt{2}\mathrm{i} \ln\left(\frac{1}{1-z}\right)\right),$$

with $J_0(x)$ the Bessel function of first kind of order 0.

The function $\psi(z)$ has a unique dominant singulary at $z = 1$; as $z \to 1$, we have

$$J_0\left(2\sqrt{2}\mathrm{i}\ln\left(\frac{1}{1-z}\right)\right)$$
$$\sim \frac{1}{\sqrt{\pi}2^{5/4}} \cdot \frac{1}{\sqrt{\ln\left(\frac{1}{1-z}\right)}} \cdot \frac{1}{(1-z)^{2\sqrt{2}}} \cdot \left(1 + \mathcal{O}\left(\frac{1}{\ln(1-z)}\right)\right)$$

Applying transfer lemmas

$$s_n = [z^n]\psi(z) \sim c \cdot \frac{n^{2\sqrt{2}-1}}{\sqrt{\ln n}} \left(1 + \mathcal{O}\left(\frac{1}{\log n}\right)\right)$$

and

$$c = \frac{3 + 2\sqrt{2}}{2^{5/4}\sqrt{\pi}\Gamma(2\sqrt{2})} = 0.8050738\ldots$$

Conclusion

We've just scratched the surface of the rich number of applications of discrete mathematics for the analysis of algorithms.

I hope I have convinced you that Analysis of Algorithms is deeply rooted in mathematics, most notably discrete mathematics.

Several of the talks in the minisymposium show nice applications of the techniques briefly described here, arising in the analysis of sorting and selection algorithms, automata theory, multidimensional data structures, polynomial factorization, decomposable combinatorial structures, union-find data structures, etc.

Other talks will present other elegant and powerful techniques that we haven't presented now.